

Estimación de Costos y Planificación de Proyectos

“Money, so they say / Is the root of all evil / Today” – Pink Floyd

"*Oh dear! Oh dear! I shall be too late!*" – White Rabbit – Alice In Wonderland

Universidad de los Andes

Demián Gutierrez

Febrero 2010



¿Cuánto cuesta desarrollar software?

-

¿Qué costos hay asociados al desarrollo de un producto de software?

Costo y el Esfuerzo



Costos de Hardware y Software

Costos de Viajes y Aprendizaje

Costos de Esfuerzo

Sueldos Ingenieros

Gastos de Seguros, Seguridad Social, etc,

Costos de Alquiler, Condominio, Luz,

Limpieza, Servicios Varios

Costos de Redes y Comunicación

Costos de Recursos Compartidos,

Administración, Salas de Reunión, etcétera



Para calcular los costos de un sistema es necesario calcular, entre otras cosas su ***tamaño*** y en consecuencia el ***esfuerzo*** necesario para desarrollarlo

¿Cómo se puede estimar el tamaño y el esfuerzo necesario para desarrollar un sistema?

Además, es necesario considerar otros ***costos indirectos*** asociados (gastos administrativos, de mantenimiento, infraestructura, equipos, etcétera)



¿métricas?

-

¿cómo mido el tamaño de una aplicación?

Métricas ¿Qué son y por qué son necesarias?



Métricas de Software: una métrica es cualquier medida o conjunto de medidas ***destinadas a conocer o estimar*** el tamaño u otra característica de un software o un sistema de información, generalmente para **realizar comparativas** o para la ***planificación*** de proyectos de desarrollo

- "Miles de Líneas de Código" (KLOC)
- Puntos de Función
- Número de Clases e Interfaces
- Errores por Caso de Uso
- Errores por Línea de Código
- **Otras...**

Fuente: http://es.wikipedia.org/wiki/Métrica_(informática)



Midiendo el Tamaño de una Aplicación...

Midiendo la Productividad de un Programador...

Cantidad de Líneas de Código



Líneas de Código

```
#include <iostream.h>

main() {
    cout << "Hello World!" << endl;
    return 0;
}
```

Hola Mundo en C++ (Aproximadamente 5 líneas de código)

```
;; Hello World for the nasm Assembler (Linux)

SECTION .data
  msg     db      "Hello, world!",0xa ;
  len     equ     $ - msg
SECTION .text
  global main
  main:
  mov     eax,4            ; write system call
  mov     ebx,1            ; file (stdou)
  mov     ecx,msg          ; string
  mov     edx,len          ; strlen
  int     0x80             ; call kernel
  mov     eax,1            ; exit system call
  mov     ebx,0
  int     0x80             ; call kernel
```

Hola Mundo en Assembler (Aproximadamente 15 líneas de código)

Cantidad de Líneas de Código



Un programa escrito en C++ tiene 500.000 lineas de código (500 ***KLOC***)

Un programa escrito en Assembler tiene 900.000 lineas de código (900 ***KLOC***)

¿Cuál de los dos programas es más grande?

¿Cuál de los dos requirió más esfuerzo?

Cantidad de Líneas de Código



Un programador escribe unas **1000** lineas de código a la semana (en Assembler)

Otro programador escribe unas **500** lineas de código a la semana (en C++)

¿Cuál de los dos programadores es el más productivo?

peor aún...

Cantidad de Líneas de Código



Un programador escribe unas **500** lineas de código a la semana (en C++)

Otro programador escribe unas **750** lineas de código a la semana (en C++)

¿Cuál de los dos programadores es el más productivo?

¿Se pueden considerar otros factores para comparar a los programadores?

por ejemplo...

Cantidad de Líneas de Código



Un programador escribe unas **500** lineas de código a la semana (en C++)
(Y posteriormente, en su código se encuentran 3 bugs)
(Escribe código difícil de entender / ilegible)

Otro programador escribe unas **750** lineas de código a la semana (en C++)
(Y posteriormente, en su código se encuentran 6 bugs)
(Escribe código fácil de entender)

Otro programador escribe unas **1200** lineas de código a la semana (en C++)
(Y posteriormente, en su código se encuentran 3 bugs)
(Escribe código fácil de entender)

¿Cuál de los dos programadores es el más productivo?



Las líneas de código en si mismas no son una métrica adecuada para medir el tamaño de un sistema

Es necesaria una métrica que sea independiente de la tecnología utilizada

Puntos de Función



Es una métrica que sirve para ***estimar el tamaño*** de una aplicación de forma ***independiente*** del lenguaje de programación o las tecnologías utilizadas

Los ***requisitos funcionales*** del sistema son ***identificados*** y ***clasificados*** dentro de cada uno de los siguientes cinco tipos: ***entradas***, ***salidas***, ***interacciones con el usuario***, ***interfaces externas y archivos utilizados por el sistema***

La métrica fue fundamentalmente diseñada para sistemas de información de gestión (de negocios, empresariales, etcétera)



Es decir:

¡Los puntos de función miden el tamaño de un sistema en términos de la cantidad de funcionalidad del sistema!

Puntos de Función (5 Componentes Básicos)



Entradas: IU -> (Archivos / BD / Otros Sistemas)

Salidas: (Archivos / BD / Otros Sistemas) -> IU

Interacciones / Consultas: IU -> Archivos / BD -> UI

Interfaces Externas: Integración con otras aplicaciones, bases de datos, etcétera externas al sistema

Archivos (Interfaces) Internos: Integración con fuentes de datos internas

Puntos de Función
(5 Componentes Básicos)



Categoría	Cantidad
Entradas	4
Salidas	3
Interacciones	4
Interfaces Externas	5
Archivos Internos	2

Se realiza una estimación, pero...
¿De dónde salen estos números?

Puntos de Función (PF)



Cada uno de los elementos de las categorías anteriores se vuelve a clasificar según su complejidad en simples, promedio y complejo, asignando pesos adicionales que van de 3 a 15 (respectivamente)

Categoría	Cantidad Total	Simples	Promedio	Complejos	Puntos de Función
Entradas	4	**1**x3	**2**x7	**1**x15	3+14+15=**32**
Salidas	3	**3**x3	**0**x7	**0**x15	9+0+0=**9**
Interacciones	4	**2**x3	**0**x7	**2**x15	6+30=**36**
Interfaces Externas	5	**3**x3	**1**x7	**1**x15	9+7+15=**31**
Archivos Internos	2	**2**x3	**0**x7	**0**x7	6+0+0=**6**
				Total FP (Function Points):	**114**

¿Quién decide la complejidad de un PF?

Puntos de Función
(Factor Ambiental / Aspectos Generales)



Se toman en cuenta las características no funcionales
(Aspectos Generales del Sistema)

	Factor Ambiental	Rating (0...5)
1	¿Se requiere comunicación de datos?	5
2	¿Existen funciones o procedimientos distribuidos?	4
3	¿Es crítico el rendimiento?	3
4	¿Se ejecutará el sistema en un entorno operativo existente y fuertemente utilizado? ¿Hay restricciones de plataforma?	0
5	¿El sistema tendrá una carga transaccional alta o baja?	5
6	Nivel de Disponibilidad	4
7	Eficiencia del Usuario Final Requerida (Usabilidad)	2
8	Actualización en Línea	2
9	Complejidad del Procesamiento	5
10	¿El sistema debe estar diseñado e implementado para ser reutilizable?	2
11	¿El sistema debe ser diseñado para ser fácil de instalar y de portar?	3
12	Facilidad de Uso	4
13	¿El sistema debe soportar múltiples instalaciones en diferentes organizaciones?	2
14	¿El sistema debe estar diseñado e implantado para facilitar cambios?	2
	Total (N)	**43**

¿Quién decide el “Rating” de cada Factor?



Se calcula el CAF
(Complexity Adjustment Factor)

```
CAF = 0.65 + 0.01 * N
CAF = 0.65 + 0.01 * 43
CAF = 1.08
```

CAF puede variar entre 0.65 (todos los ratings en 0) hasta 1.35 (todos los ratings en 5)



Se calcula el AFP
(Adjusted Funtion Points)

```
AFP = FP  * CAF
AFP = 114 * 1.08
AFP = 124
```

124 ... Bueno, ¿y qué?

Puntos de Aplicación (SLOC/FP)



Lenguaje	**SLOC/FP (Source Lines Of Code) / (Function Points)**			
--	Avg	Med	Mín	Máx
ASP	56	50	32	106
Assembler	209	203	91	320
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	66
FoxPro	36	35	34	38
J2EE (Java)	***57***	***50***	***50***	***67***
Java	55	53	9	214
JavaScript	54	55	45	63
JSP	59	-	-	-
.NET	60	60	60	60
Perl	57	57	45	60
PL/SQL	47	39	16	78

¿Quién decide cuántas SLOC por PF se van a tener por cada PF?

Fuente: http://www.qsm.com/resources/function-point-languages-table/index.html



Si estamos programando en Java

Es posible calcular la cantidad de líneas de código (Source Lines Of Code / SLOC) que tendrá la aplicación:

```
SLOC = LANG_FACTOR * PF
SLOC = 55 * 124
SLOC = 6820
```

6820 ... ¿y?

(¿Será acertado este cálculo?)

Puntos de Objeto / Puntos de Aplicación



Son métricas similares a los puntos de función pero adaptadas a otros esquemas de desarrollo / tipos de sistemas

Puntos de Objeto
(Puntos de Aplicación en COCOMO II):

Consideran:

1**) Número de pantallas independientes que se despliegan**
(Sencillas 1pto, medias 2pts, complejas 3pts)

2) Número de informes (reportes)
(Simples 2pts, moderados 5pts, complejos 8pts)

3) Módulos en lenguajes imperativos (Java / C++) que deben desarrollarse para implementar el código de acceso a la BD
(10pts cada uno)



A lo largo del proceso se han hecho las siguientes preguntas:

¿De dónde salen estos *(valores de cada uno de los componentes básicos)* números?

¿Quién decide la complejidad de un PF?

¿Quién decide el "Rating" de cada Factor?

¿Quién decide cuántas SLOC por PF se van a tener por cada PF?

¿Será acertado el cálculo de las líneas de código?



En general, para responder a las preguntas anteriores se necesita:

Requisitos...
Que permitan darle valores a los componentes 5 básicos

Un Experto (o un grupo de expertos)...
Que pueda sustentar cada una de los valores y pesos seleccionados...

Y normalmente contar con un experto o un grupo de expertos no siempre es suficiente, ya que...



... muchos de los factores que afectan estas preguntas dependen de la tecnología utilizada y de la experiencia y habilidades del equipo de desarrollo

Estadísticas...

Información de proyectos anteriores que sirva como base para realizar estimaciones confiables que tomen en cuenta los factores organizacionables no contemplados hasta los momentos



Bien, suponiendo que nuestros cálculos y estimaciones sean correctas, nuestro sistema tiene (tendrá) 124 puntos de función y si lo implementamos en Java tendrá cerca de 6820 líneas de código…

¿¿¿Y qué???



La estrategia de PF se basa en la **experiencia** del que calcule los PF (fundamentalmente) y en el conocimiento y grado de correcta aplicación de los criterios usados para calcularlos



Además...

Con los PF o la cantidad de líneas de código **no resolvemos el problema** de determinar el costo de una aplicación...



Si se puede tener una estimación del tamaño de la aplicación, entonces es posible calcular el ***esfuerzo*** requerido para desarrollar el sistema con una fórmula como la siguiente:

$$Esfuerzo = A * Tamaño^{B} * M$$

Donde el esfuerzo viene dado en P/M (Personas / Mes)



La unidad P/M es una unidad similar a las horas/hombre que sirve para calcular el esfuerzo necesario para completar una tarea

Si un proyecto toma X P/M esto significa que si se pudieran contratar X personas (en circunstancias ideales) entonces el proyecto se terminaría en 1 mes...

...o bien, significa que si sólo contratamos a una persona entonces el proyecto se terminaría en X meses

Es una unidad que permite teóricamente determinar la cantidad de personas que serían necesarias para terminar un proyecto en cierto tiempo, o la cantidad de tiempo que sería necesario para terminar el proyecto con cierta cantidad de personas

Modelado Algorítmico de Costos



A	B	M
1,05	1,05	1,02

$$Esfuerzo = A * Tamaño^{B} * M$$

Tamaño (PF)	Esfuerzo (PM)	Equipo de 5	Equipo de 10	En 6 meses	En 12 meses
PF	**$A*PF^{B}*M$**	**PM/5**	**PM/10**	**PM/6**	**PM/12**
10	12,02	2,4	1,2	2	1
20	24,88	4,98	2,49	4,15	2,07
30	38,09	7,62	3,81	6,35	3,17
40	51,52	10,3	5,15	8,59	4,29
50	65,12	13,02	6,51	10,85	5,43
60	78,86	15,77	7,89	13,14	6,57
70	92,71	18,54	9,27	15,45	7,73
80	106,67	21,33	10,67	17,78	8,89
90	120,71	24,14	12,07	20,12	10,06
100	134,83	26,97	13,48	22,47	11,24

Modelado Algorítmico de Costos



Recordar que la relación no es lineal (A*Tamaño^{B}*M)

Tiempo / Tamaño del Equipo

30
25
20
15
10
5
0

10 20 30 40 50 60 70 80 90 100

Tamaño (PF)

- Equipo de 5
- Equipo de 10
- En 6 meses
- En 12 meses

¿Cuántas personas necesito para desarrollar el proyecto en 15 días?



MR. JOHNSON DETECTS A SLIGHT PROBLEM IN JOE'S OTHERWISE FLAWLESS BUSINESS PLAN

IT SAYS HERE THAT YOU'VE HIRED NINE WOMEN SO BUSY PROFESSIONALS CAN HAVE A BABY IN JUST A MONTH.

YES SIR!

The Mythical Man Month

En general, no es posible forzar la cantidad de participantes en un proyecto más allá de cierto punto para acelerar la fecha de entrega

Lo peor que se puede hacer para resolver retrasos existentes en un proyecto es añadir mas personal al equipo de trabajo...

Ver: http://en.wikipedia.org/wiki/The_Mythical_Man-Month

COCOMO 81
(Constructive Cost Model)



COCOMO (Constructive Cost Model):
Es una estrategia basada en el modelado algorítmico de costos, desarrollada en 1981 por Barry W. Boehm

Se basó en el estudio de 63 proyectos de software desarrollados fundamentalmente usando el modelo de proceso en cascada y que oscilaban entre las 2.000 a las 100.000 líneas de código

COCOMO 81 (Constructive Cost Model)



Proyectos Simples: Aplicaciones bien entendidas desarrolladas por equipos pequeños

$$PM=2.4 * KSLOC^{1.05} * M$$

Proyectos Moderados: Aplicaciones más complejas en las que el equipo de desarrollo tiene experiencia limitada en el tipo de sistema en cuestión

$$PM=3.0 * KSLOC^{1.12} * M$$

Proyectos "Empotrados": Proyectos complejos donde la aplicación es parte de un fuerte acoplamiento de software, hardware y reglas operacionales

$$PM=3.6 * KSLOC^{1.20} * M$$

COCOMO 81
(Constructive Cost Model)



Atributos	Siglas	Valor					
		Muy bajo	**Bajo**	**Nominal**	**Alto**	**Muy alto**	**Extra**
Atributos de software							
Fiabilidad	RELY	0,75	0,88	1	1,15	1,4	--
Tamaño de Base de datos	DATA	--	0,94	1	1,08	1,16	--
Complejidad	CPLX	0,7	0,85	1	1,15	1,3	1,65
Atributos de hardware							
Restricciones de tiempo de ejecución	TIME	--	--	1	1,11	1,3	1,66
Restricciones de memoria virtual	STOR	--	--	1	1,06	1,21	1,56
Volatilidad de la máquina virtual	VIRT	--	0,87	1	1,15	1,3	--
Tiempo de respuesta	TURN	--	0,87	1	1,07	1,15	--
Atributos de personal							
Capacidad de análisis	ACAP	1,46	1,19	1	0,86	0,71	--
Experiencia en la aplicación	AEXP	1,29	1,13	1	0,91	0,82	--
Calidad de los programadores	PCAP	1,42	1,17	1	0,86	0,7	--
Experiencia en la máquina virtual	VEXP	1,21	1,1	1	0,9	--	--
Experiencia en el lenguaje	LEXP	1,14	1,07	1	0,95	--	--
Atributos del proyecto							
Técnicas actualizadas de programación	MODP	1,24	1,1	1	0,91	0,82	--
Utilización de herramientas de software	TOOL	1,24	1,1	1	0,91	0,83	--
Restricciones de tiempo de desarrollo	SCED	1,23	1,08	1	1,04	1,1	--

Fuente: http://es.wikipedia.org/wiki/COCOMO

COCOMO 81
(Constructive Cost Model)



Modelo Básico:

M = 1

Modelo Intermedio:

M = RELY * DATA * CPLX * TIME * STOR * VIRT * TURN * ACAP * AEXP * PCAP * VEXP * LEXP * MODP * TOOL * SCED



COCOMO II (Constructive Cost Model): Desarrollado por Barry Boehm y otros autores en el año 1997 (publicado en el año 2000). Incorpora a COCOMO 81 elementos adicionales que permiten hacer mejores estimaciones en función a las técnicas y tecnologías de desarrollo de software existentes en la actualidad

COCOMO II
(Constructive Cost Model)



La página oficial de COCOMO II:
http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

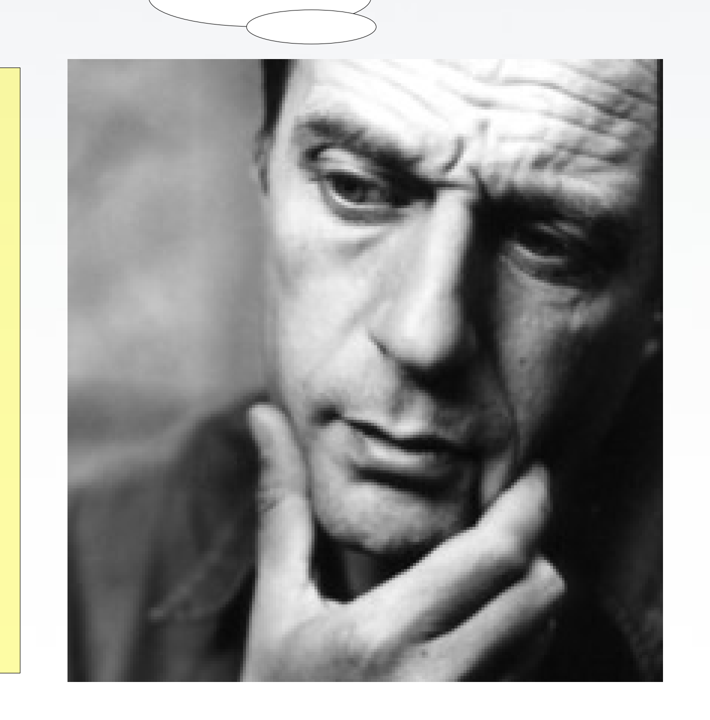
Herramienta para hacer estimación con COCOMO y con otros esquemas de Modelado Algorítmico de Costos:
http://www.softstarsystems.com/

Estimación por Analogía



Proyecto	SLOC	Esfuerzo (PM)	Costo (BsF)	Páginas Documentación	Errores	Defectos	Personas	Tiempo de Desarrollo (Meses)
Alfa	12.100	24	45.000	165	134	29	3	8,00
Beta	27.200	62	140.000	245	321	86	5	12,40
Gamma	20.200	43	150.000	500	256	64	6	7,17
...	...	...	...	...	...	...	...	...

El costo se calcula por comparación con proyectos similares en el mismo dominio de aplicación (Métricas)





Ventajas:

Preciso si se dispone de datos previos

Desventajas:

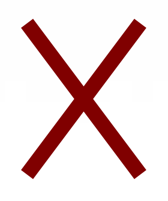
Imposible de realizar si no se han desarrollado proyectos comparables.

Es necesario mantener una base de datos con la información necesaria.

Es necesario que las condiciones generales de los proyectos a comparar sean similares.



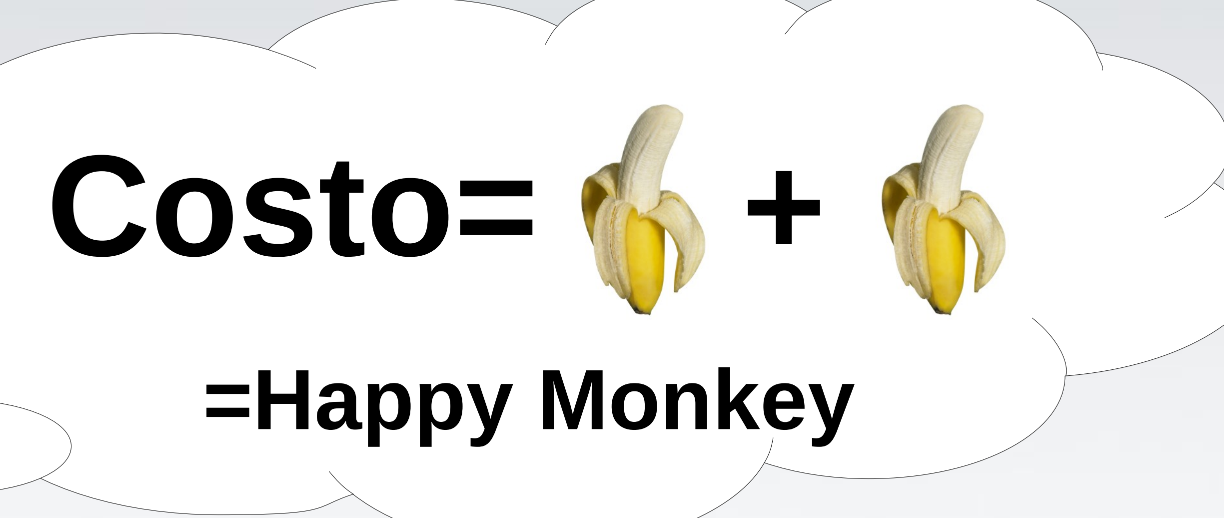
Uno o más ***expertos***, tanto en ***desarrollo de software*** como en el ***dominio de la aplicación*** usan su experiencia para ***predecir los costos*** del software.
Se realizan ***iteraciones*** hasta llegar a un ***consenso***.

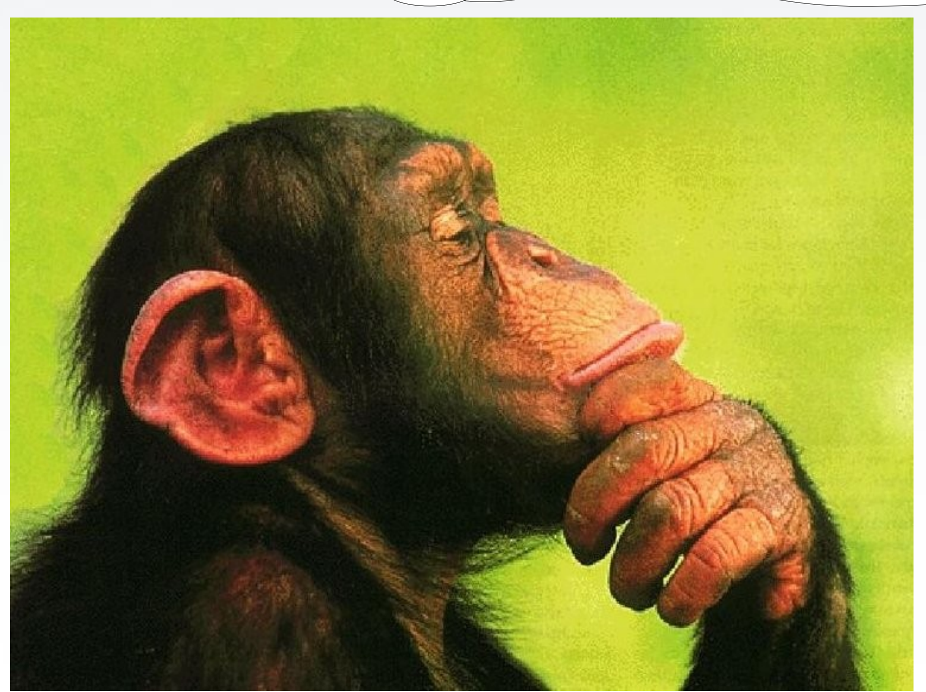


Siguiente... El experto...

Juicio Experto







<== *El Experto!!!*



Costo = 200.000 BsF



De verdad, no es broma...

¿Cómo lo hacen?

Juicio Experto



CU01
actor
CU02
CU09
CU03
CU10
actor
CU04
CU05
CU11
actor
CU06
CU07
CU12
actor
CU08

El costo se puede estimar en función a un grupo de requisitos / casos de uso / funcionalidad inicial, la experiencia de los expertos en el dominio y la experiencia de los expertos en software

Juicio Experto



Tiempos / Costos de Desarrollo "Sistema XXX"

Caso de uso / Concepto	Tiempo (Días)	Costo
	Costo / hora	**Bs65,00**
Arranque del proyecto		
Levantamiento de Requerimientos / Refinar casos de uso	15	Bs7.800,00
Arquitectura del sistema / arranque del proyecto	10	Bs5.200,00
Subtotal	**25**	**Bs13.000,00**
Recursos Logísticos		
Flujo Incorporación a Inventario	2	Bs1.040,00
Flujo Desincorporación a Inventario	2	Bs1.040,00
Buscar Desincorporación a Inventario	1	Bs520,00
Inventario	3	Bs1.560,00
Subtotal	**8**	**Bs4.160,00**
Control de calidad / ajustes finales		
Control de Calidad	20	Bs10.400,00
Instalación y puesta a punto final	5	Bs2.600,00
Subtotal	**25**	**Bs13.000,00**
Manuales / Entrenamiento		
Desarrollo del manual de usuario	20	Bs10.400,00
Entrenamiento	0	Bs0,00
Subtotal	**20**	**Bs10.400,00**
Total (Tiempo neto en días / Costos)	**78**	**Bs40.560,00**

Una hoja de cálculo es su mejor amigo al momento de calcular costos...

Juicio Experto



Ventajas:

Suele ser muy barato.

Puede ser muy exacto si se cuenta con los expertos adecuados.

Desventajas:

Imposible de realizar (o muy impreciso) si no se cuenta con los expertos adecuados



Los costos del proyecto están en función de los recursos disponibles, utilizando todo el tiempo permitido

(Normalmente esto no es una buena idea)

Ventajas:

No realiza presupuestos abultados

Desventajas:

El sistema normalmente no se termina

(O se desperdicia tiempo / recursos)



El costo del proyecto está en función de lo que el cliente está dispuesto a pagar
(Normalmente esto no es una buena idea)

Ventajas:
La empresa de software consigue el contrato (desarrollar un producto que vale 50.000 BsF por tan sólo 5.000 BsF)

Desventajas:
La probabilidad de que el cliente obtenga el trabajo es mínima ya que los costos no reflejan verdaderamente el trabajo requerido

Costos fijos



Concepto	Costos Fijos	
Administradores	2.000,00	Incluye seguro y otros aspectos legales
Limpieza	600,00	Incluye seguro y otros aspectos legales
Alquiler	2.000,00	
Papelería	1.000,00	
Luz / Agua	400,00	
Redes / Comunicación	500,00	
Condominio	200,00	
TOTAL (A)	**6.700,00**	

No olvide calcular y contemplar los costos fijos de algún modo... y si puede consiga un administrador que se encargue de los detalles financieros

Sueldo Ingeniero (B)	4.000,00	
Cantidad de Ingenieros (C)	15,00	
Costos Fijos / Ingenieros (D)	446,67	A/C
Costo Mensual / Ingeniero (E)	4.446,67	B+D
Costo Hora/Hombre (F)	26,47	E / (21 días) / (8 horas)

Recordar también los costos de depreciación de equipos...

¿Requerimientos antes de costos?



Muchos Ingenieros de Software coinciden en:

...la necesidad de profundizar en la especificación de requerimientos (o tener algún tipo de requerimientos) ANTES de ejecutar la estimación de plazos y esfuerzos del proyecto de software...

Simplemente, ¿cómo se puede calcular el costo de un edificio sin saber cuantos pisos se van a construir?



¿Planificación y Gestión del Proyecto?

¿Requerimientos antes de costos?



Es necesario realizar la ***planificación*** y ***calendarización*** del proyecto.

Esto se logra ***definiendo y desglosando*** las ***tareas*** que hay que realizar para poder llevar a término el proyecto.

Luego se define ***quién*** realizará las tareas, ***qué*** recursos se necesitarán, ***cuándo*** se realizaran y ***cuál*** será el orden en el que se realizaran.

Planificación (Tareas)



Proceso Técnico 1

Proceso Técnico 2

Grupo de Procesos

Proceso Técnico N

NO ENTRAR EN MUCHO DETALE

Proceso A

...

Proceso B

...

Proceso C

...

Subproceso A.1

...

Subproceso A.n

...

Fuente: Adaptado de GRAY WATCH MÉTODO DE DESARROLLO DE SOFTWARE PARA APLICACIONES EMPRESARIALES / Noviembre 2008

Planificación (Hitos)



Estudio de Viabilidad → Perfilar Requisitos → Desarrollo de Prototipos → Especificación de Requisitos

- Estudio de Viabilidad ↓ Informe de Viabilidad
- Perfilar Requisitos ↓ Definición de Requisitos
- Desarrollo de Prototipos ↓ Prototipos; Informe de Evaluación de Prototipos
- Especificación de Requisitos ↓ Especificación de Requisitos

La definición y verificación de Hitos es una forma de darle "visibilidad" al proceso

NO ENTRAR EN MUCHO DETALE

Fuente: Adaptado de EPS Informática UAM (T3: 18/19)

¿Requerimientos antes de costos?



El desglose de tareas se realiza en base al ***proceso*** (actividades generales a realizar) y en base a las tareas específicas resultantes de los requerimientos (funcionalidad)

Identificación de Tareas:
(2.2.3.2)
2 Driver de Pantalla.
3 Utilidades visualiz.
2 Diseño.
2 Desarrollo.

NO ENTRAR EN MUCHO DETALE

WBS del proyecto
Sistema de control y de comandos

1 Gestión y Administración
2 Desarrollo
3 Soporte al Desarrollo

2.1 Análisis
2.2 Diseño
2.3 Codific.
2.4 Pruebas

2.2.1 Lógica de Control
2.2.2 Interfaz de comandos
2.2.3 Utilidades visualiz.
2.2.4 Comunicaciones

2.2.3.1 Formato visualizac.
2.2.3.2 Driver de Pantalla
2.2.3.3 Generador de Formas

¿Requerimientos antes de costos?



Proyecto de Prueba - /home/dmi/Dropbox/IngDelSoftB2009/clase_6_costos/proyecto_de_prueba.pod

OPENPROJ File Edit View Insert Tools Project Help

No Filter | No Sorting | No Group

	Name	Duration	
1	Sistema XXX	41.333 d...	2/2
2	Requisitos	19.333 d...	2/2
3	Iteración 1	4.333 da...	2/2
4	Definir la Visión del Sistema	0.333 d...	2/2
5	Definir Requisitos	1 day	2/2
6	Describir Requsitos	2.5 days	2/2
7	Validar Requsitos	0.5 days	3/2
8	Iteración 2	6.5 days?	3/3
9	Realizar Modelo de Dominio	0.5 days?	3/3
10	Realizar Diagrama de CU General	1 day	3/3
11	Realizar Descripción General de CU	2.5 days	3/4
12	Refinar Visión del Sistema	0.5 days?	3/9
13	Refinar Requisitos	1 day	3/9
14	Validar CU Generales	0.5 days?	3/1
15	Validar Requisitos	0.5 days?	3/1
16	Iteración 3	8.5 days?	3/1
17	Refinar Diagrama de CU	1 day	3/1
18	Refinar Descripción General de CU	2.5 days	3/1
19	Refinar Visión del Sistema	0.5 days?	3/1
20	Refinar Requisitos	1 day	3/1
21	Validar CU Generales	0.5 days?	3/1
22	Validar Requisitos	0.5 days?	3/1
23	Preparar DDR y DER	2 days	3/1
24	Entrega Final DDR y DER	0.5 days	3/2
25	Definir Arquitectura Preliminar	22 days?	3/3
26	Investigar Arquitecturas Existentes	7 days	3/3
27	Seleccionar Arquitectura Base a Utilizar	7 days	3/1
28	Puesta a Punto del Ambiente de Desarrollo	7 days	3/2
29	Arquitectura / Frameworks Definida y Prob	1 day?	4/1
30	Diseño Arquitectónico	15 days?	4/2
31	Iteración 1	13 days?	4/2
32	Especificar Arquitectura General	1 day?	4/2
33	Especificar Arquitectura Detallada	7 days	4/5
34	Definir el Modelo de Datos	5 days	4/1
35	Iteración 2	2 days	4/2

Feb 10 | 28 Feb 10 | 7 Mar 10 | 14 Mar 10 | 21 Mar 10 | 28 Mar 10 | 4 Apr 10

Pedro Perez;Jose Paredes
Jose Paredes;Pedro Perez
Juan García
Juan García;Jose Paredes;Pe
3/24
4/2

Modelos Incrementales (Modelo Incremental)



Requerimientos y Funcionalidad

Método / Proceso

Recursos humanos

Planificar sólo pensando en los requerimientos o sólo pensando en el proceso (especialmente esto último) usualmente no es adecuado

La ***planificación*** y el ***cálculo de costos*** son problemas muy complejos con muchas variables, pero en general, las más importantes son los métodos y procesos usados, los requerimientos y los recursos humanos

¿Otros aspectos de la gestión de proyectos?

Una reflexión final sobre lo profundo del abismo



- Gestión de Grupos (**eGroupware**, PHPGroupware)
- Gestión de Código Fuente (SCM) (**CVS**, **SVN**, Hg, Git, etc)
- Gestión y Asignación de Tareas (Redmine, **dotProject**, **eGroupware**, Trac)
- Manejo de Clientes (CRS)
- Manejo de Contenido (CMS) (**Drupal**, Joomla, etc)
- Gestión Ágil de Proyectos (SCRUM, XP, Otros)
- Soporte al Usuario (Foros, Issue Trackers y otras herramientas
- Gestión de Bugs (**FlySpray**, Trac, Jira, **Mantis**, etc)
- Gestión de Recursos Humanos
- Base de Conocimiento (Wiki / Otras) **Dokuwiki**, Mediawiki, etc
- Discusiones (Foros y otras herramientas) (**mybb**, phpbb, **fluxbb**, SMF)
- Gestión de Documentos (**GoogleDocs**, **KnowledgeTree**, **eGroupWare**, Collanos)
- **otras...** (Launchpad, Atlassian (tools), PHProjekt, Gforge, Saros)

Algunas Herramientas de Planificación



Planner (Libre/Escritorio):
http://live.gnome.org/Planner

Openproj (Libre/Escritorio):
http://openproj.org/

Dotproject (Libre/Web):
http://www.dotproject.net/

M$ Project (Propietario/Escritorio)

Otros...



Más información en:
http://www.pmi.org/
(PMBok: Project Management Body of Knowledge)

Otras herramientas interesantes para revisar



http://marketplace.eclipse.org/content/saros-distributed-collaborative-editing-and-pair-programming



Ultimately, everyone wants estimates with the highest possibility or probability of being correct. If an estimate has an equal amount of possibility to be early as late, then this is the highest probability. Imagine a bell curve for a moment. At the peak of the bell curve (the mean value) the most likely outcome. This should be the estimate. If, on the other hand, an estimate has no probability of being early, then it has just about zero percent possibility of actually happening and about 100% probability of being late. In other words, the estimate is to the far left side of the bell curve. A good estimate should have an equal probability of being early or late.

When an estimate is based upon some quantitative process, it is easier to stand behind the estimate. When the estimate is based upon a guess, there is more likelihood to cave when pressure is applied. The reason for this is simply a lack of confidence in the estimating process. The only thing to change about an estimate is the inputs used to generate an estimate. If the estimate is too high, then functionality needs to be reduced.

Esto está genial para futuros cursos

Fuente Reboot Rethink /
http://www.rebootrethink.com/forcesBehavior/miracleWorkers.php



The reason **overtime is rewarded** is because software development managers **have no other metric** in place **to measure performance**. These same managers **illogically conclude** a person **who is working overtime** is a **dedicated, productive, and hard working** employee.

Crudo pero cierto en muchos casos...

¿Recuerdan las 40 horas a la semana de XP?

Fuente Reboot Rethink / http://www.rebootrethink.com/forcesBehavior/workingOvertime.php



Estimates need to be non-negotiable. An estimate should be created using a quantified method. That means there is some method to creating estimates. Put some data into a formula and derive the result. The only thing you should be willing to budge on is the inputs. There are several inputs with an estimate including size of the project, deadlines, staff, so on and so forth. Hence, if the estimate is too high, then one of the inputs needs to be changed.

Unfortunately, what traditionally happens is that an estimate is nothing more than a guess. The estimate has no substance at all; in other words, it is not based upon historical performance or statistical modeling. Often when working on a contract I ask the question, “How did you come up with your estimate?” More often than not the person actually admits it was a guess. Another common answer begins, “based upon my vast experience as a software professional.” In other words, questioning the estimate is the same as questioning their integrity. It is important for an estimator to be able to quantitatively explain how they derived their estimate.

Fuente Reboot Rethink /
http://www.rebootrethink.com/forcesBehavior/nonNegotiable.php


UNIVERSIDAD
DE LOS ANDES
VENEZUELA

¡Gracias!


Backspace
Success
Shift